THAT WHICH IS CLAIMED:

1. A system for testing an application running on a target device, the system comprising:

a target device storing and executing a software test agent, wherein said application under test is also stored and executed on said target device;

a test development computer storing and executing a software test tool for testing and validating said application's rendering of output to a Graphical User Interface (GUI), said test tool configured to communicate with said agent on said target device;

said software test tool executing a test script for testing and validating one or more aspects of said application's rendering of output to said GUI, said test tool operable to generate requests to said agent to obtain information from and send events to controls associated with said GUI; and

said software test tool comprising a trap manager for handling trap events that can occur on said target device during execution of said test script, said trap manager executed by said test development computer to:

allow a user to define a trap event and create an associated user-defined function for handling the trap event;

automatically check whether the trap event has occurred during execution of said test script; and

upon detecting an occurrence of the trap event, execute said user-defined function to handle said trap event.

2. The system of claim 1, wherein said trap event is a window that appears during execution of said test script, said window being defined by the user according to one or more window properties that can be checked by the agent to determine whether said window is actually the trap event.

3. The system of claim 1, wherein said test tool comprises a capability to turn on and off during the execution of said test script, the checking of a particular trap event.

4. The system of claim 1, wherein said test tool comprises a capability to turn on and off during the execution of said test script, the checking of all trap events.

5. The system of claim 1, wherein said test tool comprises a capability that allows a user to assign an input-event name to a grouping of multiple key sequences, said grouping of multiple key sequences representing an input event that occurs on said target device during execution of said test script, wherein said input-event name can be used in place of said grouping each time said grouping is to be written into said test script.

6. A method of handling a trap event during execution of an automated test of an application running on a target device, the method comprising:

creating a user-defined function for handling an occurrence of the trap event;

defining the trap event using a software test tool being executed by a test development computer;

checking to see whether the trap event has occurred on said target device during execution of said automated test; and

upon detecting an occurrence of the trap event, executing said user-defined function to handle the trap event.

7. The method of claim 6, wherein said trap event is a window that appears during execution of said automated test, said window being defined according to one or more window properties that can be checked to determine whether said window is actually the trap event.

8. A method of handling a trap event during execution of an automated test of an application running on a target device, the method comprising:

creating a user-defined function for handling an occurrence of the trap event;

defining the trap event using a software test tool executing on a test development computer;

storing trap event data on the target device, said data capable of being used by an agent executing on the target device to detect the occurrence of the trap event;

monitoring with the agent to determine whether the trap event has occurred on the target device;

upon detecting an occurrence of the trap event, transmitting a notification of the occurrence of the trap event from the agent to the test tool, the notification comprising a user-defined name for uniquely referring to the trap event;

accessing a table on the development computer using said trap event name to obtain a pointer that points to the user-defined function for handling the trap event; and

executing the user-defined function to handle the trap event.

9. A system for testing an application running on a target device, the system comprising:

a target device storing and executing a software test agent, wherein said application under test is also stored and executed on said target device;

a test development computer storing and executing a software test tool for testing and validating said application's rendering of output to a Graphical User Interface (GUI), said test tool configured to communicate with said agent on said target device;

said software test tool executing a test script for testing and validating one or more aspects of said application's rendering of output to said GUI, said test tool operable to generate requests to said agent to obtain information from and send events to controls associated with said GUI; and

said software test tool comprising a configuration manager for handling testing of said application against multiple languages and platform configurations.

10. The system of claim 9, wherein the configuration manager comprises a configuration table that includes one or more configurations, each said configuration comprising a collection of value sets, each said value set comprising a collection of one or more related configuration items and corresponding values for said related items.

11. The system of claim 10, wherein the configuration table is stored in a spread sheet format for ease of editing by a user.

12. The system of claim 10, wherein, responsive to a user changing from a first value set to a second value set during execution of a test script, the test tool is configured to automatically delete all configuration item values associated with the first value set and reload said configuration items with corresponding values associated with the second value set.

13. The system of claim 10, wherein the configuration manager comprises a capability that allows a user to get the value of a particular configuration item, and a capability that allows a user to set the value of a particular configuration item, during execution of a test script.

14. The system of claim 9, wherein said test tool comprises a capability that allows a user to assign an input-event name to a grouping of multiple key sequences, said grouping of multiple key sequences representing an input event that occurs on said target device during execution of said test script, wherein said input-event name can be used in place of said grouping each time said grouping is to be written into said test script.

15. A method of testing an application that is operable to execute in multiple languages and platform configurations, the method comprising:

storing a configuration table having a plurality of user-defined configurations, each configuration comprising a collection of value sets, each said value set comprising a collection of one or more related configuration items and corresponding values for said related items;

writing a test script that executes differently based on which user-defined configuration is loaded from said table;

loading a user-defined configuration from said configuration table prior to execution of said test script; and

executing said test script.

16. The method of claim 15, wherein the configuration table is stored in a spread sheet format for ease of editing by a user.

17. The method of claim 15, further comprising:

changing from a first value set to a second value set during execution of said test script; and

responsive to said changing:

automatically deleting all configuration item values associated with the first value set; and

automatically reloading said configuration items with corresponding values associated with the second value set.

18. A system for testing an application running on a target device, the system comprising:

a target device storing and executing a software test agent, wherein said application under test is also stored and executed on said target device;

a test development computer storing and executing a software test tool for testing and validating said application's rendering of output to a Graphical User Interface (GUI), said test tool configured to communicate with said agent on said target device;

said software test tool executing a test script for testing and validating one or more aspects of said application's rendering of output to said GUI, said test tool operable to generate requests to said agent to obtain information from and send events to controls associated with said GUI;

said software test tool comprising a configuration manager for handling testing of said application against multiple languages and platform configurations; and

said software test tool comprising a trap manager for handling trap events that can occur on said target device during execution of said test script, said trap manager executed by said test development computer to:

allow a user to define a trap event and create an associated user-defined function for handling the trap event;

automatically check whether the trap event has occurred during execution of said test script; and

upon detecting the occurrence of the trap event, execute said user-defined function to handle said trap event.

19. A method performed by a test development computer storing and executing a test tool operable to communicate with a target device storing and executing a test agent to interrogate a particular Graphical User Interface (GUI) control that is associated with an application under test running on said target device, the method comprising:

storing a first table that comprises a list of entries, each said entry containing information corresponding to a particular GUI control associated with said application under test, the information associated with each said control comprising:

a user-defined name for uniquely referring to that control;

a plurality of properties which define that control, including a class name that indicates a class type to which that control belongs; and

a data field for optionally storing a control-specific identify flag, said control-specific identify flag used for indicating which of said properties are to be used in identifying that control on said target device;

storing a second table that comprises a list of entries, each said entry containing information related to a particular class of GUI controls associated with said application under test, the information associated with each said class of controls comprising:

a class name for uniquely referring to that class of controls; and

a default identify flag for indicating which of said properties are to be used for identifying any control of that class type that does not have a control-specific identify flag associated with it, as defined in said first table.

20. The method of claim 19, further comprising:

locating in said first table the user defined name of said control to be interrogated;

reading from said first table the properties associated with said control to be interrogated;

checking to see whether the control-specific identify-flag data field associated with said control is empty;

loading the identify flag from said control-specific identify flag data field, if said data field is not empty;

if the control-specific identify-flag field is empty, loading the default identify flag from the second table using the class name associated with said control to be interrogated;

generating a request to the agent to locate a control on the target device that matches said control to be interrogated, the agent locating a matching control by searching for a control which has a set of properties indicated by the identify flag obtained in the steps above that match the corresponding set of properties for said control to be interrogated, the matching control having a handle for accessing said control associated therewith; and

receiving said handle for said matching control from the agent.

21. The method of claim 20, wherein the entry associated with each said control listed in said first table further comprises a data field for optionally storing a control-specific verify flag, said control-specific verify flag used for indicating which of said properties are to be used when verifying that an occurrence of that control on said target device matches an expected state; and

wherein the entry associated with each said class of controls listed in said second table further comprises a default verify flag, said default verify flag used for indicating which of said properties are to be used for verifying any control of that class type that does not have a control-specific verify flag associated with it, as defined in said first table

22. The method of claim 21, further comprising:

checking to see whether the control-specific verify-flag data field associated with said control to be interrogated is empty;

loading the verify flag from said control-specific verify flag data field, if said data field is not empty;

if the control-specific verify-flag field is empty, loading the default verify flag from the second table using the class name associated with said control to be interrogated;

interpreting the verify flag obtained in the steps above to determine which properties are to be used in verifying that said matching control matches said control to be interrogated;

generating a request to the agent to retrieve actual property values associated with said matching control, said values retrieved comprising values for at least those properties indicated in the verify flag obtained in the steps above; and

comparing said actual property values associated with said matching control, for only those properties indicated in the verify flag obtained above, to the corresponding property values defined in said first table for said control to be interrogated.

23. The method of claim 22, further comprising:

upon detecting a mismatch in any of the properties compared, generating an entry in a log file detailing the mismatch.